FE EDUCATION CENTER

KINGSTON, NEW YORK

The attached material was developed for use in forums on 2065 BCU which were conducted in Area 4 by Mr. Jack Cronwell, Senior FE Specialist, and has been reproduced for your information.

All of the attached material was developed by Jack. Because he wanted to be able to refer to specific AND's, OR's, and triggers in his write-up, he developed his own second level diagrams from logic. You should be aware that many of these second levels are contained in the present 2065 FEDM (Y27-2038-0) and in the 2065 Revision Material (Chapter 2, Section 3).

3-13-69

BCU
or
I Love a Mystery

(A short novel by Jack Cronwell
Area 4 - TAG)

This is a package which I have put together for 2065 men who have a basic understanding of the function of BCU. "BCU is a synchronizing circuit designed to assign storage to contending users - those users being CPU, and channels." The basic reason to have such a circuit is obvious to a trained 2065/67 man, but "how it works" seems to be a "mystery". This package will take away that mystery.

It consists of a second level which shows the BCU circuitry involved for a request per the "D" reg. in CPU to an even address. This can also be used as an excellent tool when shooting BCU problems. I have done so many times. It also consists of the following write-up in which I lead you through the second level, circuit by circuit, (each "and", "or", and "trigger" is numbered) from the request trigger going on to the final "BCU clean-up" pulse, which is a result of that request.

Notes:

A) Abbreviations
   1) Triggers = Tx's
   2) Latches = LTH's
   3) Circuit = CIRC

B) Number for the CIRC's are in a circle in the lower right side of the logic block.

C) "And/or" block's inputs are differentiated by parenthesized letters (ie., (a), (b), etc.), as in "and/or" CIRC. #7.

BCU-I

1) "D sync. Tx " - This is turned on in CPU by the MREQ*D micro-order. It will be turned on at P4 of the clock time of the ROS block in which you see the micro-order.

2) "D sync. LTH" - This is turned on at not clock of the same cycle in which the sync. Tx comes on. This has an additional turn on, which is not shown, but it could be turned on directly by the "NEOP" or IFETCH "reset" micro-order hardware circuitry. This second turn on is necessary since those circuits cannot be brought up in time to turn on the Tx and so must turn on the LTH. in order to get the request in as soon as possible.

   The output of this feeds the OR, circuit #3.

3) This "or" has 4 inputs the "D synch LTH" "IC synch LTH" and "scan synch LTH", and the CPU request Tx which is actually a feedback circ. which will hold on to the fact that we are making a CPU request if we do not get the select out immediately. It has two main outputs. One goes off and starts the CPU sequencers which will stop the CPU clock if our request is not honored immediately by the next clock time. I will discuss their operation later in another write up. The other output attempts to get our request into BCU with "and" circuit #4.

4) This "and" will allow us to try to turn on our priority Tx if the "BCU busy Tx" is not on, which would say that some other request has not yet finished using BCU circuits.

5) "BCU busy Tx" - This trigger goes on whenever someone gets priority to use the BCU circuits, for channels the BCU response line comes up for any channel that is given access to the BCU circuits. This Tx will not go off until BCU clean-up comes, which will be at the end of a successful BCU operation when a select pulse is actually sent off to some storage.

6) "CPU priority Tx" - This Tx going on will say that a CPU request has control of the BCU circuitry. Even though the line, "turn on CPU priority" is up, we cannot turn the Tx on if the address valid LTH is on. This is a LTH that comes on early in the channel/BCU selection operation so that the channel would still take priority over us. if this is not so though, we do get priority.

Yea! Yea! we just got into BCU. Nay, Nay, though varlet, thou must yet get thee a select pulse sent to thy storage. If thou shouldst not get this, thou wilt be hung in BCU with thy priority Tx on forever and ever, as thou wilst not get "BCU clean upst". Yea! verily, let us see if we can even "try" to send our select. *

7) "And/or" #7 has an output, which will turn on the storage 2 Tx. This output is called "Issue a select". STG 2 Tx going on generates a pulse which we may or may not send out to a storage. One of the inputs to the CIRC, "and" (a) #7, is the one which CPU will bring up to first turn on the STG 2 Tx.

*TRANSLATION-Even though we have control of BCU at this point, if conditions are not right (i.e. Invalid address, storage busy, etc.) we will not be able to send a select pulse to our storage!

One of the other inputs to cir. #7, "and" (b), is conditioned by a channel getting priority, which is essentially the channel "address valid lth". The two "and's" just discussed will only be active in the beginning of the operation as they will be deconditioned once STG. 2 Tx turns on STG. 2 LTH. The 3rd. "and" (c) is conditioned if the 1st. select pulse we just generated was not actually "sent" to the storage and would continue to force pulses until a pulse was actually sent to some storage. This could result if the storage unit were busy or we were requesting an invalid address.

8) "STG. 2 Tx" - This Tx going on generates the select pulse. The BCU oscillator is timed with this Tx going on to generate a 135 nanosecond pulse which we may or may not try to send to the storage depending on the validity of the address.

   The STG. 2 Tx has a second turn on besides "and/or" #7; the STG. 1 LTH. which is only used with an LCS attached to the system. Note that the Minus "LCS advance waiting line is tied to +6V".

9) "STG. 2 LTH" - This LTH always goes on at not clock of the cycle that the Tx goes on. It will prevent the turning on of the STG. 2 Tx by "and's" (a) or (b) in "and/or" #7.

   STG. 2 LTH will also turn on STG. 3 Tx if a BCU cleanup does not immediately come on for this pulse which will in turn, turn on the STG. 2 LTH again. We will continue this loop until we get a BCU clean-up for this request.

10) "And" Circ. #10 output will be the timed select pulse. It is taken to all the possible storage unit circuits in the BCU, and if the address

on the SAB's is a valid one to one of the storages, we will "try" to send it to that storage.

BCU-II

11)"And" circ. #11 is one of several, each of which is in the separate frame select circuits. If the power is on that frame, the select pulse is allowed to go further on into the frame circuitry.

12)"And" #12 is in the address decode circuitry which determines if the SABs contain a valid address to an existing memory box. This is determined by pluggable cards at installation time. It's output is either odd or even address decode depending on the defeat or no defeat interleave function and SAB 6 or 20. Note that one leg of this "and" also requires CPU priority. There are equivalent ones for channel requests, odd and even select.

13)At this point we are sitting at the door of "and/or" #13. At this point, we have gotten into BCU, generated a select pulse, determined that we have power on our frame, and that we have a valid address on the SABs. Note that if we had not had a valid address or power on the frame, we cannot get through "and/or" circuit #13, "and(a)". This is a very critical point, since it is here where an invalid address is truly detected.

The output of #13 is called "high speed select tried 1, I". You must realize that when you see "select tried" circuitry, this means the address tried was valid, and to a frame with power. Whether the select goes out or not has to do with that frame's availability but you do know that it is a good address at this point.

Also notice, at "or" #14, the inputs are labeled 1, I; 1, II; 2, I; 2, II; etc. This type labeling is also used in "select sent" circuitry. The first No. refers to the physical frame (box) and the second No. (roman numeral I or II) refers to the odd or even select in that box. I = even and II = odd.

I wish to emphasize that when you see "select tried" line labels you know it means a valid select is being tried and "select sent" labeling means a select pulse was actually sent out to a frame!

14) "HSS sel tried Tx" - Turned on for any frame's sel tried line and prevents Invalid address Tx's from coming on. !!!!

At this point, let's discuss a good, healthy request for an even, valid address, power on frame type select. Nothing but clean living for us, guys!

15) "And/or" #15 is again one of several for each possible frame and odd/even portion of the frame. The output of this will send our select pulse on a simplex line to the specific storage we want, where it will become the basic clock pulse. (Oh, BCU, I love you!) Notice that "and (b)" has all the same conditions on it as "and/or" circuit 13", "and (a)", plus the fact that the storage must not be "busy" finishing a former select.

16) "And" #16 has an output which comes up if we send out our select called "pseudo accept". Well in this case pseudo "ain't" so pseudo since this is the only way the BCU realizes that he did send a select and can now do a BCU clean-up operation. A 2365 does not send back an accept pulse!!

17) "And" Circ. #17 has the same inputs, essentially, (valid address, frame power, select pulse and not busy) as "and/or" #15 ________ ________. The output is labeled "select sent frame 1, I". This output is or'ed together with all other "select sent" lines and turns on what is essentially the "select sent Tx" - "or" 18 and "and" 19. Now friends in radio land, keep calm, but guess what happens ....

20) Good old "and" circuit #20 has one leg conditioned "select sent" and the other is labeled "HSS early accept." Now you know this came up at not B1 time as the select pulse went to our storage back at "and" #16 and went through the "or" #21A. Leapin' lizards sandy, the output of "and" 20 is --- BCU clean-up!!! (I may cry). This good line and "and" 21 (late BCU clean-up) will reset; HSS select sent Tx (and #19); HSS select tried Tx (14), prevent STG 3 Tx from coming on again so we won't generate another select pulse and reset "BCU busy Tx" #5. Good, good, good; now the next guy can get a chance at BCU.

Oops, one other small item men. What if our request had been for an invalid address, or a frame with power off? Notice that we would not get through "and/or" 13 (Select tried circuit) since "and" #12, (invalid address) or; "and"#11, (power off frame) would not be active. Well, let's see what this does to "and's" 23 and 24.

22) "Test for invalid address Tx" goes on every cycle after the STG 2 LTH goes on. (Essentially, our select pulse). This conditions one leg of our "and's" 23 and 24. If we haven't turned on our HSS select tried Tx #14, the outputs go up and we notify the CPU of the condition if this a CPU request and; turn on the invalid address Tx's 1 and 2. These triggers will <u>never</u> go on unless a request is made to an invalid address, or a frame with power off!! Notice that we're kind of stuck here though, as without HSS select sent, 18; we can't get a BCU clean-up to let anyone else in BCU and do anything about the invalid select! Not so my friends. We get another select pulse at "and/or" #7, and (c). And we will send this pulse to the lowest frame with power, "and/or" #15, and (a) due to the invalid address Tx being on. This gives us an accept pulse and "and/or" #13, and (c) will come up giving us the select tried, and at this point we can get our BCU clean-up. One other point of interest on the inv. address, select, we send cancel over with our select which will force 0's with good parity on the MDBO so if this were a fetch, we will ingate good parity to the

register waiting for the data and will not get a bad parity, machine check condition which would confuse the actual "inv. address prog. check" condition.

Well boys and girls, Annie and Sandy made it through the terrible BCU safely. But what happened to Daddy Warbucks and Punjab when they went off into the land of the evil CPU sequencers? That you'll know if you pick up the next episode, "Punjab and the CPU-stop-clock-trigger".

Jack Cronwell
Sr. FE Specialist - Area 4 - TAG

BCU Problems I Have Known and Loved

or

Punjab and the CPU-Stop-Clock-Trigger

(sequel to "I Love a Mystery")

The CPU-Stop-Clock-Trigger has been the key to every "BCU Bug" I have shot in the 2065. These are the times when you are called to the machine because it has stopped and you see the "Inhibit Clock" trigger is on. Someone says over your shoulder "Oh, oh, it's a BCU problem", and your blood runs cold! There is no need for this reaction if you read and fully understand the following write up and service aid. You should also have read my previous BCU write up. (An aspirin also helps)

Whenever CPU makes a memory request per the "D", "IC", or "Scan" circuits; as it tries to get priority, it also starts the "CPU sequencers" BCU III, "or" #3.

These sequencers are used to tell BCU how long it has been since he made his last memory request; CPU must know this for very good reasons. If for some reason CPU does not get his request honored immediately, his clock must be stopped. This is done so that:

1) He will still have the correct address available in the register making the request.

2) He will not try to ingate data before it is available on the busses on a fetch,

There are three reasons that he may not get his request honored immediately if you will recall.

1) BCU is busy with another request.
2) A higher priority request is contending for BCU.
3) The storage, CPU is requesting is busy.

In all the cases CPU's request will not be sent out as a select immediately, and until it is, he will not get a "BCU Clean-up" sent to CPU. (Help keep our BCU clean).

You will see later that it is this pulse, BCU clean up, which either prevents the CPU-Stop-Clock-Trigger from setting, or resets it to start the clock again when the select is finally sent out. I will devote the majority of this write-up to the setting and resetting of this trigger since it has been the key to the BCU problems I have fixed. (Let's not talk about the others).

But first, let me discuss the sequencers. They are essentially a trigger to latch to trigger operation once they have been started. There are two basic types of requests, a three and a four cycle type. Looking at BCU III, you will see that at "and/or" circuits #'s 33 and 35, if the "three cycle" trigger is on, the sequencers operate as follows:

CPU 2 trigger
CPU 2 latch
CPU 3 trigger
CPU 4 latch
CPU 5 trigger
CPU 5 latch

If the "three cycle" trigger is off, (a four cycle request has been made) the sequence is

CPU 2 trigger
CPU 2 latch
CPU 3 trigger
CPU 3 latch
CPU 4 trigger
CPU 4 latch
CPU 5 trigger
CPU 5 latch

You can see the first sequence was three cycles long, and the second was four.

It is when the CPU 2 latch comes on that we may be allowed to turn on the "CPU-Stop-Clock" trigger at "and/or" #38 "and" a! You must understand one fact. If when we make our request, the BCU sends our request out immediately, CPU receives "BCU clean-up" on the very next cycle, that is; <u>the same cycle the CPU-2 trigger and latch comes on</u>!!

Now, let's look at the "CPU-Stop-Clock" trigger, "and/or" #39. If we can ever get one leg on each of the "or's" a, b, and c minus at one time, we can get the trigger on. Let's take the clock cycle just after the CPU-2 latch came on. He will remain on until the next not clock PO. It is at this time plus PO that we will turn it on if at all.
At "or" a, the top leg will be plus at clock time.
But , the 2nd leg will be minus most of the time as it will not go plus until we get the CPU 5 latch <u>and</u> the advance pulse. Well, this will not happen until the end of the storage cycle. This is used with an LCS only when the clock is stopped a second time with the CPU 4 latch to await data from the LCS.

At "or" b, the top leg is used with the insert key function. Let's discuss the normal select, so this will be plus. The next leg down is the one which will be minus since "and/or" 38, "and" a has the CPU 2 latch and PO since we are talking of the next cycle after the latch went on.

Now, if we can get one leg of "or" c up, we will turn on the trigger. The top leg is, "+BO or B2". We are at clock time, so this is plus, the second leg is "-insert key" and we are not doing this so this will be plus.

The third leg is "+CPU priority". This could be the leg which would turn us on if BCU had been busy to us and we did not get priority, but if we had gotten priority, this leg would be plus, so we would have to look at the last leg.

This fourth leg, "+BCU clean-up successful" is the deciding leg. If we had not received BCU clean up immediately, you can see that on this cycle, this leg would be minus and we would turn the trigger on, it would remain on until CPU did receive BCU clean up. "Or" c would only have this leg minus if you examine the other legs. (CPU priority is not reset until late BCU clean up!) When this "BCU clean up successful" goes plus, the trigger will reset.

Realize again, that if BCU clean up had been there due to the fact that we had gotten our select sent to storage, the trigger would not have even been set.

If for some reason CPU makes too many requests too fast to BCU, the symptom is usually the fact that the "CPU-Stop-Clock" trigger goes on and does not get reset. This turns on the "inhibit clock" trigger and that's what you see on the console

"Here come the judge"!!

Now, let's talk about the bug which hangs CPU as above.

There are certain basic problems which we must overcome before we can shoot this at all.

1) Recognize the problem as being of this nature.

2) Get ourselves some kind of a restart after the hang and some kind of loop for scoping.

3) Find some place to look and sync with the scope, which will get us back to the reason for the clock stopping and not restarting.

The first problem is quickly solved as soon as you realize that the CPU is stopped with the "inhibit clock" trigger on. Look at roller 3, Position 1. The second is the matter of two tie downs and some research into the problem.

One tie down will give you a reset every 16 milliseconds. It takes the output of the interval timer single shot and puts it into the switch circuits such that any switch which you jam in will be functional every 16 ms. Realize that reset is an overriding function and that, if you had another switch jammed in, you would not see it while resetting. This reset gets us to ROS 003 and when it is gone, we will be in the stop loop. This tie down is 01E-E2F7B3 (KWO11) to 01A-C2D6D4(KD601).

Now we will jam in a second buttom with another tie down. This tie down will bring up a line called "push button gate". With this up, any button which is looked at in the stop loop, and jammed in will be functional when we go by that ROS block in the stop loop. We'll jam load PSW. The tie down is to ground 01C-B4 G03 D09.

Now every 16 ms. we will get a reset, and then when the reset disappears, we will be in the stop loop and when we get to the ROS block for LPSW, we'll do it. This is our restart after the hang up. Of course, we must now find a good PSW restart point to go back to to get our hang. A good thing to do is to look at IC to find where you are, get a listing and go back about four or five instructions before. Here I must leave you to your own devices as the hang condition could be due to many things occurring. (i.e., you could have branched here, etc.)

The third problem is what this is all about, the "CPU-Stop-Clock" trigger. He's the place to start at. Scope the output, and you will see him going on and off. Then it will go on and stay on until the reset occurs. That's the point of time you're interested in!! Put the other probe on the output of "or" #3 BCU III, MC068. IC, D or scan synchs latches. I can almost guarantee that you'll see two requests coming in one clock cycle after another. Now you must investigate them and you'll probably find that one is extraneous.

At this point, remember that many requests are generated by hardware at end OP time and I fetch time. They will always turn on the "D" or "IC" request latch instead of the trigger. The micro orders enter the triggers first. I have found such things causing the problems as:

1) Hot IC 21, 22 = 01, or 10 lines or D 21, 22 = 01, or 10! This causes incorrect hardware requests per IC or D to refill "Q".

2) Wild ROS branches. Taking me to a block which makes a request just after a good hardware request has been made.

3) Hot lines up to some "and's" on KD201 which is where all the branches make their hardware requests to refill "Q" per the "D" or "IC", during I fetch with the "reset" micro-order.

One thing you can be sure of - CPU is never allowed to make one request right after another without at least one cycle in between!

Notice that what we have called "BCU problems" have really been in the CPU, but resulted in stopping the clock due to BCU's inability to handle extraneous requests.

Summary of BCU problem approach.

1) Determine some LPSW return point to restart machine. (about 4 instructions before failing point)

2) a. tie down: 01E-E2F7B3 to 01A-C2D6D4
and; 01CB4G3D9 to ground
b. jam: "reset" button
"LPSW" button

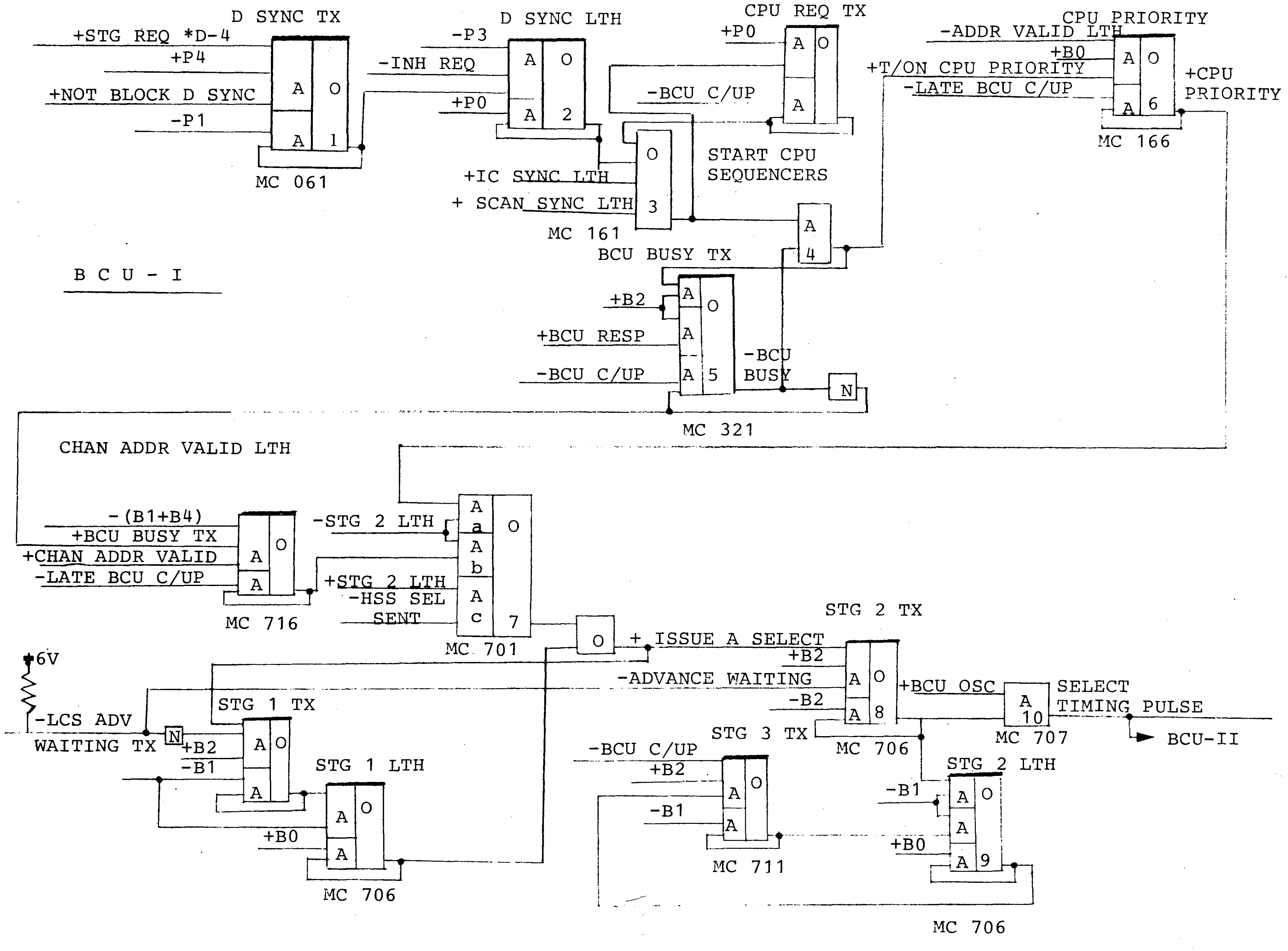
3) Scope output of CPU-Stop-Clock trigger. Note point where it goes on and does not reset.

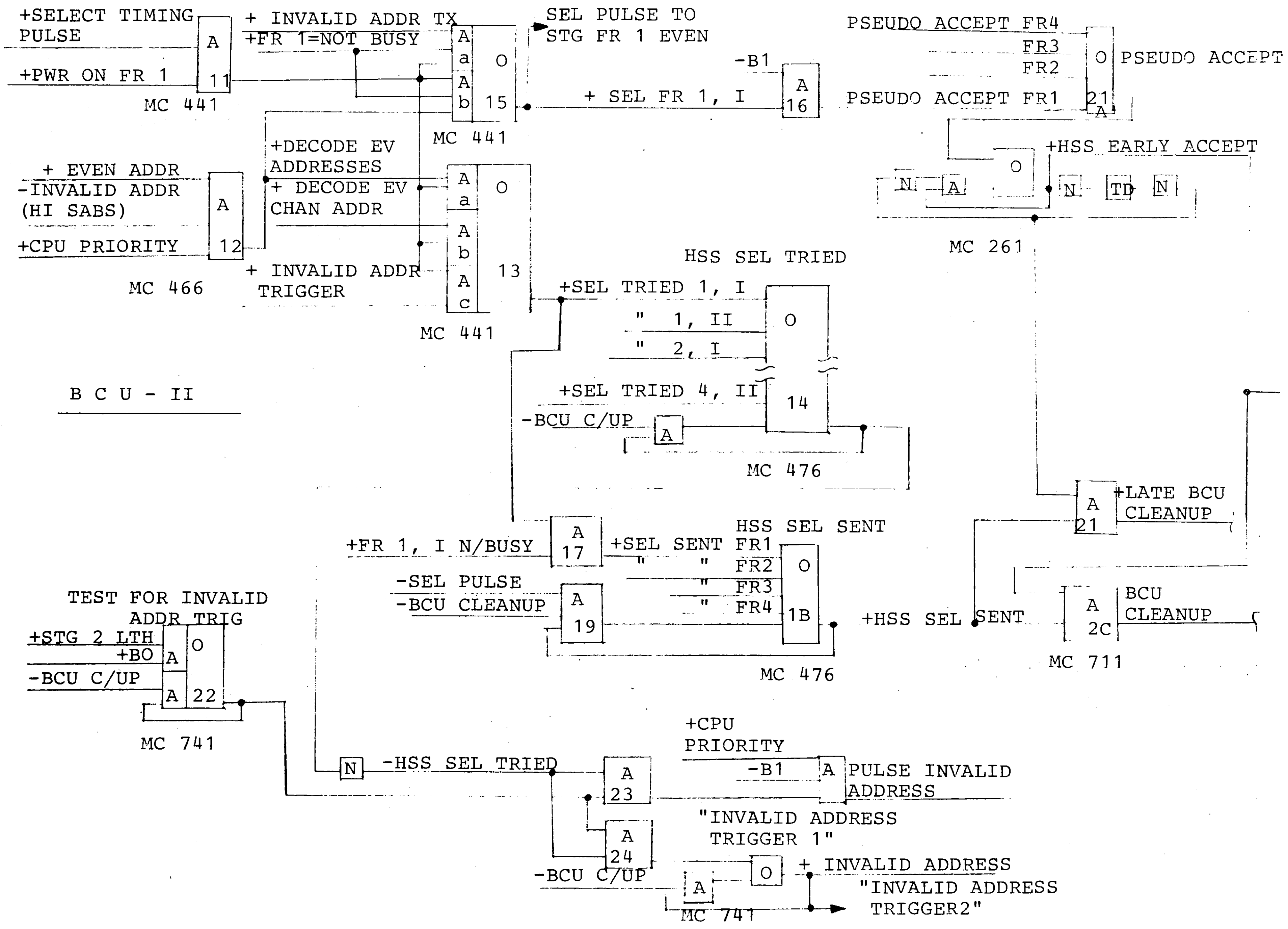
4) Look at requests being made at this point of time from "or" #3 on MC-068. Investigate both of these. One will be right, and one will be in error.

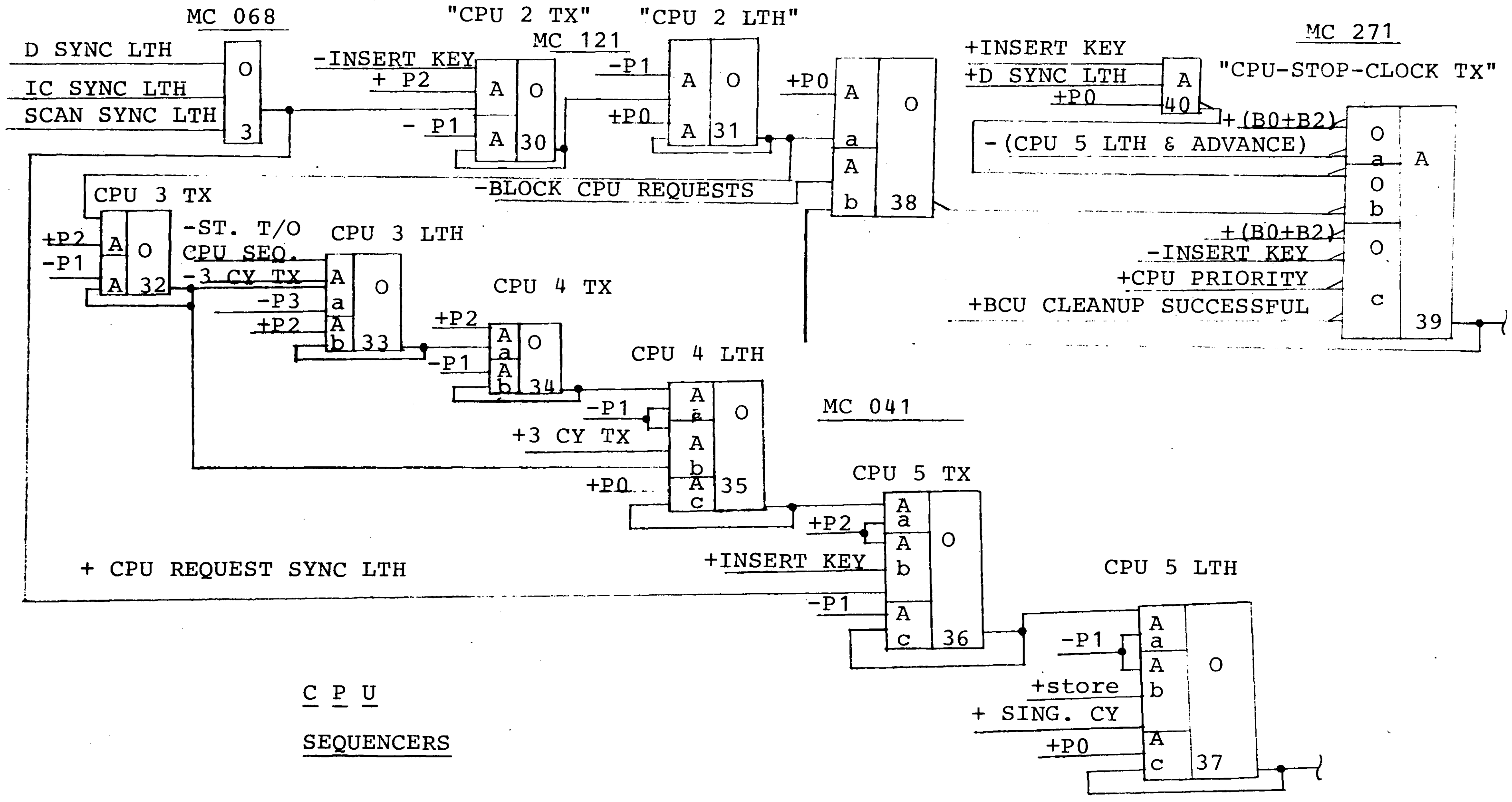
So then kids, punjab has beaten the "evil CPU sequencers". But will he, Daddy Warbucks and Annie ever meet again? I think you'll be surprised if you tune in next week when you'll hear Annie singing to Sandy, "Sitting on the console, watching all the bits go by".

Don't forget to send in 3 CPU covers and we'll send you the magic hexadecimal decoder along with your handy dandy ROS plane torque wrench.

Jack Cronwell
Senior FE Specialist


D SYNC TX
+STG REQ *D-4
+P4
+NOT BLOCK D SYNC
-P1
A
O
A
1
MC 061
D SYNC LTH
-P3
-INH REQ
+P0
A
O
A
2
CPU REQ TX
+P0
A
O
A
-BCU C/UP
O
3
+IC SYNC LTH
+ SCAN SYNC LTH
MC 161
START CPU SEQUENCERS
A
4
CPU PRIORITY
-ADDR VALID LTH
+B0
+T/ON CPU PRIORITY
-LATE BCU C/UP
A
O
A
6
+CPU PRIORITY
MC 166
B C U - I
BCU BUSY TX
+B2
+BCU RESP
-BCU C/UP
A
A
A
O
5
-BCU BUSY
N
MC 321
CHAN ADDR VALID LTH
-(B1+B4)
+BCU BUSY TX
+CHAN ADDR VALID
-LATE BCU C/UP
A
O
A
MC 716
-STG 2 LTH
+STG 2 LTH
-HSS SEL SENT
A
a
A
b
A
c
O
7
MC 701
O
+ ISSUE A SELECT
STG 2 TX
+B2
-ADVANCE WAITING
-B2
A
O
A
8
MC 706
+BCU OSC
A
10
MC 707
SELECT TIMING PULSE
BCU-II
+6V
-LCS ADV WAITING TX
N
STG 1 TX
+B2
-B1
A
O
A
STG 1 LTH
+B0
A
O
A
MC 706
STG 3 TX
-BCU C/UP
+B2
-B1
A
O
A
MC 711
STG 2 LTH
-B1
+B0
A
O
A
A
9
MC 706


+SELECT TIMING PULSE
+PWR ON FR 1
A 11
MC 441
+ INVALID ADDR TX
+FR 1=NOT BUSY
A a A b
O 15
MC 441
SEL PULSE TO STG FR 1 EVEN
-B1
+ SEL FR 1, I
A 16
PSEUDO ACCEPT FR4
FR3
FR2
PSEUDO ACCEPT FR1
O 21 A
PSEUDO ACCEPT
+HSS EARLY ACCEPT
N A O N TD N
MC 261
+ EVEN ADDR
-INVALID ADDR (HI SABS)
+CPU PRIORITY
A 12
MC 466
+DECODE EV ADDRESSES
+ DECODE EV CHAN ADDR
+ INVALID ADDR TRIGGER
A a A b A c
O 13
MC 441
HSS SEL TRIED
+SEL TRIED 1, I
" 1, II
" 2, I
+SEL TRIED 4, II
O 14
-BCU C/UP
A
MC 476
B C U - II
+FR 1, I N/BUSY
A 17
+SEL SENT FR1
HSS SEL SENT
" " FR2
" FR3
" FR4
O 1B
-SEL PULSE
-BCU CLEANUP
A 19
MC 476
+HSS SEL SENT
A 21
+LATE BCU CLEANUP
A 2C
BCU CLEANUP
MC 711
TEST FOR INVALID ADDR TRIG
+STG 2 LTH
+BO
-BCU C/UP
A A O 22
MC 741
N
-HSS SEL TRIED
A 23
A 24
-BCU C/UP
+CPU PRIORITY
-B1
A
PULSE INVALID ADDRESS
"INVALID ADDRESS TRIGGER 1"
A O
+ INVALID ADDRESS
MC 741
"INVALID ADDRESS TRIGGER2"



C P U

SEQUENCERS

B C U - III

BCU FUNCTIONAL UNITS

1. STG. 1 TX- KEEPS TRYING TO TURN ON STG. 2 IF (LCS) ADVANCE WAITING TX IS ON
2. STG. 2 TX - GENERATES A SELECT PULSE TO TRY TO SEND TO STORAGE FRAME.
3. STG. 3 TX - HOLDS ON STG. 2 LTH. IF SELECT DID NOT GET SENT. (INVALID ADDRESS, OR BUSY STORAGE)
4. BCU BUSY TX - GOES ON WHEN A PRIORITY IS ESTABLISHED, GETS RESET ON BCU CLEAN-UP. SAYS BCU IS BUSY TRYING TO SEND A SELECT.
5. HSS SELECT TRIED TX - SAYS A SELECT WAS SENT OUT OR WAS TRIED TO A BUSY STORAGE UNIT. DOES NOT COME ON FOR INVALID ADDRESS OR NO POWER.
6. HSS ACCEPT LTH - TURNED ON BY PSEUDO ACCEPT WHEN SELECT IS SENT OUT. THE LAST THING TO GO ON WHICH GENERATES BCU CLEAN-UP.
7. BCU CLEAN-UP - COMES ON SHEN A SELECT IS SUCCESFULLY SENT TO A FRAME.
8. INVALID ADDRESS TX 1 - SAYS A SELECT WAS ATTEMPTED AND THERE WAS NO HSS SELECT TRIED TX.
9. CANCEL - FORCED ON WITH INVALID ADDRESS TO GATE 0'S ON SDBO! SENT TO STORAGE TO THE LOWEST BOX WITH POWER WHEN INVALID ADDRESS FORCED A SELECT TO THIS BOX.
10. CPU SEQUENCERS - DETERMINES HOW FAR CPU HAS GONE AFTER EACH STORAGE REQUEST.
11. STOP-CPU-CLOCK TX - TURNS ON TO STOP CPU SO WE CAN HOLD AN ADDRESS FOR SAB & DO NOT GATE SDBO TOO SOON.
12. CPU-2 LTH - USED TO TURN ON STOP-CPU-CLOCK IF REQUEST DOES N OT GET HONORED IMMEDIATELY.
13. CPU-4 LTH - USED WITH LCS TO STOP CPU WHEN LCS ADVANCE DOES NOT ARRIVE IN TIME SO WE WAIT TO GATE IN THE STORAGE BUSSES UNTIL ADVANCE ARRIVES.

TROUBLE SHOOTING BCU ON 2065

SYNCHING -

1. THE BEST SYNC IS THE ROS BLOCK ON WHICH THE REQUEST TO STORAGE IS MADE. (MS-REQ*D-3) FOR EXAMPLE) OR IN THE END OP BLOCK IN WHICH THE NEOP OR BEOP MICRO-ORDER MAKES A REQUEST PER THE IC TO REFILL THE Q WHEN IT IS NECESSARY. (REFERENCE 2065 HANDBOOK, PAGE 33, ALD KD101, KD201)

2. YOU CAN ALSO SYNC ON THE PARTICULAR REQUEST TRIGGER IF YOU KNOW IT ONLY COMES ON FOR THE OPERATION YOU ARE CONCERNED WITH.

WHAT TO LOOK FOR ---

GENERALLY, YOU'RE LOOKING FOR THE SELECT PULSE TO THE BOX YOU'VE SELECTED AND BCU CLEAN-UP.

SELECT PULSE TO::

BOX 1 I (EVEN)	- MC441
BOX 1 II (ODD)	- MC442
BOX 2 I (EVEN)	- MC443
BOX 2 II (ODD)	- MC442
BOX 3 I (EVEN)	- MC444
BOX 3 II (ODD)	- MC445
BOX 4 I (EVEN)	- MC446
BOX 4 II (ODD)	- MC445

WHILE SYNCHING EXTERNALLY ON THE REQUEST, LOOK ON THE "A" PROBE AND :

1) CHECK TO SEE IF YOU GET CPU PRIORITY (PAGE MC166)
2) SEE IF YOU HAVE STG 2 TRIGGER UNDER THE PRIORITY. (PAGE MC706)
3) KEEPING STG 2 TX ON CHANNEL "A" SEE IF YOU GET YOUR SELECT PULSE TO THE STORAGE BOX ON CHANNEL "B" PROBE, 1, 2, OR 3 CYCLES LATER.

THE ONLY REASON YOU SHOULDN'T GET IT IMMEDIATELY IS STORAGE BUSY. MAKE SURE YOU DO NOT GET INVALID ADDRESS (PAGE MC741) BEFORE THE SELECT PULSE, UNLESS IT IS AN INVALID ADDRESS YOU ARE REQUESTING!!!